SimMPI: An MPI-blockset for Simulink

Gaurav Sharma[1], Dr. A Krishnamurthy[1], B. L. Esken[2], T. Menke[3]
SharmaG@ece.osu.edu, AKK@ece.osu.edu, Bruce.L.Esken@saic.com,
Timothy.Menke@wpafb.af.mil

[1]Electrical and Computer Engineering, Ohio State University, Columbus, OH
[2] Science Applications International Corporation, Beavercreek, OH
[3] SimAF, WPAFB, OH

Abstract

In this paper we present SimMPI, a Simulink blockset that enables insertion of MPI function calls in a graphical Simulink model and hence allows automatic generation of *parallelized* C-code and executables directly from a Simulink model. Two sample studies are presented. The first is a simple Monte Carlo estimation of π, the second is a complex high fidelity radar model. In both cases, the parallelized model showed significant performance gains.

1. Introduction

The Mathworks Inc.'s Simulink provides a GUI based environment for modeling a wide variety of complex dynamic systems. The systems are modeled by a block diagram containing a hierarchical set of blocks that represent components or subsystems of the main system. This diagram represents various time-dependent mathematical relationships between systems' inputs, outputs and states [2]. Real time workshop (RTW) is an extension to Simulink that allows creation of stand-alone applications via automatic code generation and compilation from Simulink models [1], essentially removing the effort of designing and maintaining various data structures while creating portable and efficient code.

Message Passing Interface (MPI) standard provides for portable, easy to use libraries for message passing programs [5]. Several implementations such as MPICH, LAM-MPI, and MPVAICH (infiniband interconnects) are now available.

While simple message passing programs are quite easy to create and maintain, same cannot be claimed true for programs that model complex real world problems. In fact, for highly complex models parallel computation paths can be extremely difficult to

determine by simple code analysis. At the same time, GUI based modeling approach can lead to easier identification of such computation paths.

Several Simulink block sets for cluster computing exist, e.g. Opal-RT's RT-Lab [3], and Extessy AG's DS (Distributed Simulation) Toolbox [4]. However, their widespread use can be and is hampered by factors such as, proprietary libraries, requirements for specialized hardware/software platforms, and specialized applications.

2. SimMPI: A Simulink Block-set

The SimMPI block library was developed with aims similar to the MPI Standard – to provide a practical, easy to use, efficient and flexible library for creating parallelized models in Simulink. The block library enables a user to insert MPI calls directly in the graphical model, thereby allowing automatic generation of parallelized C-code with properly inlined MPI function calls.

2.1 Current capabilities:

a. Currently, SimMPI supports several MPI (e.g., basic send and receive, broadcast) and MPE (e.g. MPE_Decomp1d) functions. Some of the SimMPI blocks are shown in Figure 1. These can simply be dragged and dropped into a Simulink model.

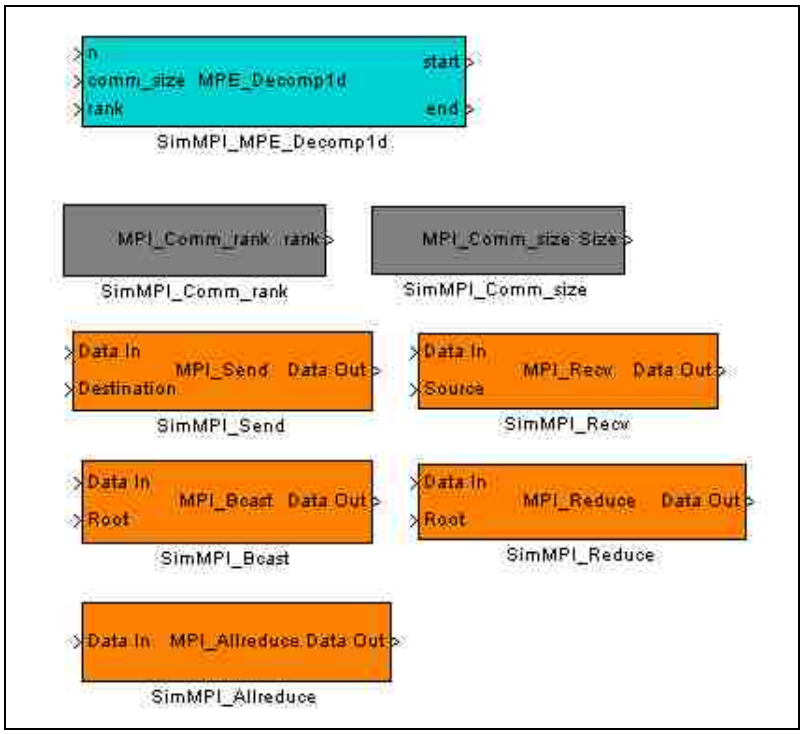

Figure 1: Some SimMPI Blocks

b. During code generation, the SimMPI blocks automatically map the Simulink datatypes to the MPI datatypes. Currently, only the basic C-datatypes are supported. Note, that this limits the ability of each communication block to handle only a single datatype at a time. We intend to provide support for multiple datatypes soon.

c. The SimMPI blocks dynamically adjust for the number of data items, and manage the message tags without requiring any input from the user. At the same time the library also provides blocks where these parameters can be specified by the user. For blocks such as the MPI_Allreduce block the user specific parameters (e.g., for MPI_Allreduce the MPI Operation) can easily specified through a graphical user interface.

d. MPI enabled C-code and executables for both Windows and UNIX environments can be generated directly from the Simulink environment using the usual RTW interface (Figure 2).

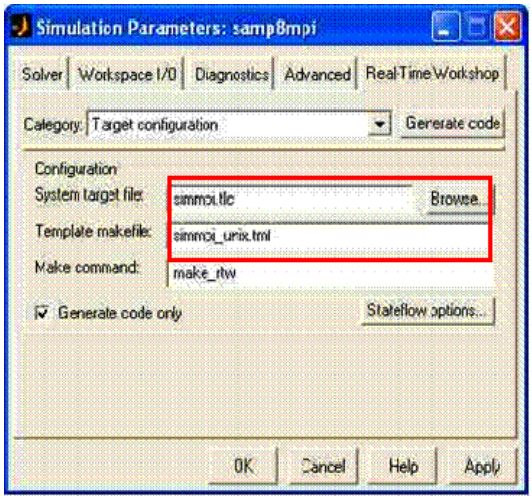

Figure 2: SimMPI simulation parameter specification

3. Experiments and Results

In this section, two sets of experiments that demonstrate the capabilities of a small subset of the MPI block library are described briefly.

a. Monte Carlo estimation of π:

This simple application, adapted from [5], demonstrates the capability of SimMPI blocks for MPI_Allreduce and MPI_Comm_rank.

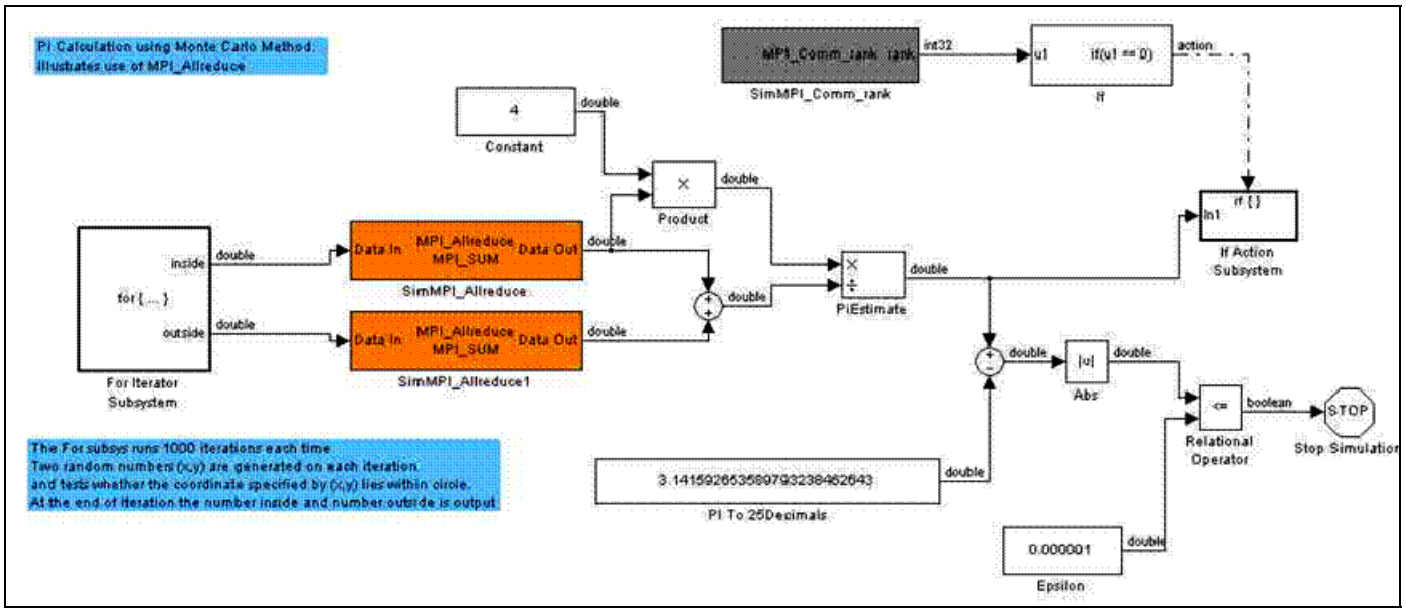

Figure 3: Monte Carlo Estimation of π

Briefly, $\boldsymbol{\pi}$ is estimated by generating random (x, y) points in a unit square and testing whether these points lie within the square's in-circle and using the fact that the ratio of the in-circle to the area of the square is $\boldsymbol{\pi}/4$. Each loop generates 1000 random sets of (x, y) pairs. The ratio of number of points in and out of the in-circle provides an estimate of the value of $\boldsymbol{\pi}$. The simulation terminates when the difference between the $\boldsymbol{\pi}$-estimate and a preset value of $\boldsymbol{\pi}$ falls below a specified limit *epsilon*.

Note the MPI_Allreduce block now shows the collective MPI operation (MPI_SUM) to be performed (Figure 3). Our simulation runs show the following time estimates for *epsilon* = 10^{-10} (Table 1).

Number of Processors	Time (seconds)	Gain
1	212	1
2	121.31	1.75
4	57.73	3.67
6	37.12	5.71

Table 1: Simulation times for Model 1 (*epsilon* = 10^{-10})

b. A Complex Radar model:

Figure 4 shows a Radar model which consists of four main component systems – target model, environment model, radar model, and analysis controller. Each of these is a highly complex model with several layers of hierarchical subsystems.

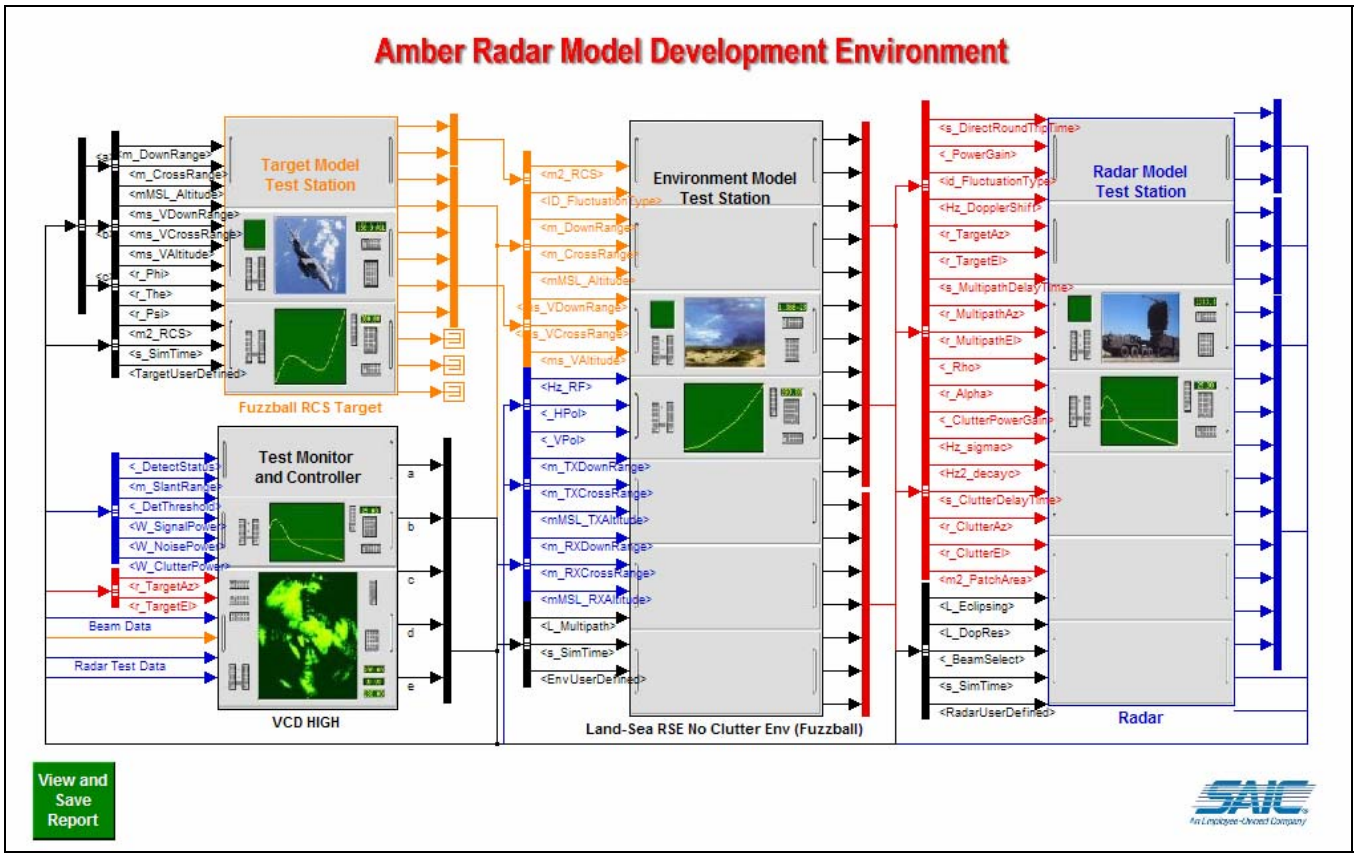

Figure 4: A Radar Model

The radar model in the current experiment models a 36 beam position 3D steered radar array system operating in a no clutter environment with a single target.

A simple parallelization scheme that split the calculation corresponding to each beam on a different processor was implemented. This was achieved using a block arrangement shown in Figure 5.

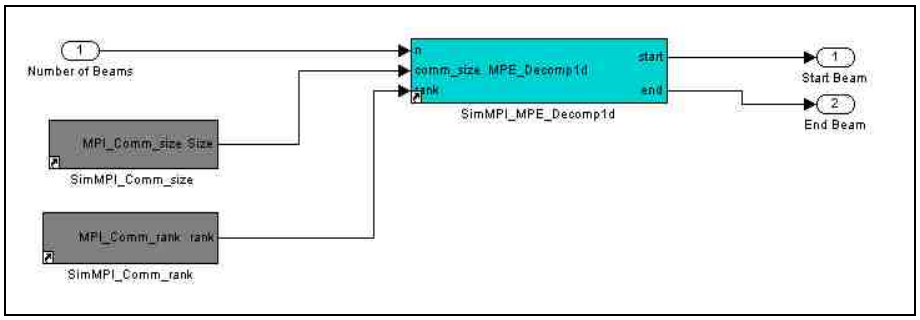

Figure 5: Task decomposition across processors

The MPE_Decomp1d function computes a balanced decomposition of a 1-D array. The performance gains are listed in Table 2.

Number of Processors	Time (seconds)	Gain
1	1513	1
2	792.72	1.91
4	399.70	3.78
6	271.06	5.58

Table 2: Simulation times for Model 2

4.Conclusion

A new Simulink blockset for creating parallel models was presented. A strong case can be made for the utility of this blockset for a wide range of applications from the two sample applications presented – a 'toy' Monte Carlo estimation of $\boldsymbol{\pi}$ and a parallelized high fidelity radar model. We believe that this blockset will find immense use with industrial users.

5. References

[1]. Mathworks' User Guide and Reference Manual: Real-Time Workshop

[2]. Mathworks' User Guide and Reference Manual: Simulink

[3]. Opal-RT's RT-Lab 6.2 User Guide and Reference Manual

[4]. Web reference: Extessy AG, Germany: www.extessy.com

[5]. William Gropp, Ewing Lusk and Anthony Skjellum, 'Using MPI: Portable Parallel Programming with the Message Passing Interface', The MIT Press, Cambridge, USA, 2nd ed., 1999.